



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|-----------------|-------------|----------------------|----------------------|------------------|
| 10/603,303 | 06/25/2003 | Hung T. Nguyen | 02-5604 (4028-03000) | 3040 |

24319 7590 05/17/2005

LSI LOGIC CORPORATION
1621 BARBER LANE
MS: D-106
MILPITAS, CA 95035

| |
|----------|
| EXAMINER |
|----------|

DANG, KHANH

| | |
|----------|--------------|
| ART UNIT | PAPER NUMBER |
|----------|--------------|

2111

DATE MAILED: 05/17/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/603,303

Applicant(s)

NGUYEN ET AL.

Examiner

Khanh Dang

Art Unit

2111

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☐ Responsive to communication(s) filed on ____.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-43 is/are pending in the application.
- 4a) Of the above claim(s) ____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) ____ is/are allowed.
- 6) ☒ Claim(s) 1-43 is/are rejected.
- 7) ☐ Claim(s) ____ is/are objected to.
- 8) ☐ Claim(s) ____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on ____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. ____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date ____.
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. ____.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: ____.

DETAILED ACTION

Claim Rejections - 35 USC § 112

Claims 24 and 43 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

In claim 24, the phrase "the device [peripheral device] comprises an interrupt control unit coupled between the peripheral device and the processor and configured to receive the first interrupt signal from the peripheral device" (emphasis added) is unclear.

In claim 43, it is unclear what may be "the third and fourth stages of the instruction execution pipeline." The third and forth stages are not defined in claim 40.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1, 2, 4-12, and 20-24 are rejected under 35 U.S.C. 102(b) as being anticipated by Heberle.

Art Unit: 2111

As broadly drafted, these claims do not define any structure that differs from Heberle (5,237,322)

With regard to claim 1, Heberle discloses a processor (in Heberle, the master unit M, capable of sending data to or read data from at least one slave unit S, is readable as a processor) configured to execute an instruction comprising a user-defined value, wherein the user-defined value is either an address or a command (at least a predetermined address in Heberle), and to provide the user-defined value during execution of the instruction (it is clear that in Heberle, the master M comprising the address generating section provide a predetermined address during execution of the instruction).

With regard to claim 2, it is clear from Heberle that the address comprising address bits is predetermined, or in other words, the address bits are assigned by a user.

With regard to claim 4, the predetermined value wherein the user-defined value is an address of an addressable register (the register of the slave S storing an address of the slave corresponding to the predetermined value).

With regard to claim 5, it is clear that the register storing the address of the slave S is accessed in response to the predetermined address generated by the master).

With regard to claim 6, it is clear that the processor of Heberle provides a predetermined value or the user-defined value, data corresponding to the user-defined value or data transmitted to a particular addressable slave, and an asserted write control signal during execution of the instruction (the processor or master can perform

Art Unit: 2111

read/write instructions, and wherein the data is stored in the addressable register in response to the user-defined value (the address of the slave is stored in a register of the slave), the data corresponding to the user-defined value (it is clear that the data corresponds to the predetermined address), and the asserted write control signal (the read/write from and to the master/processor).

With regard to claim 7, the read/write instruction comprising predetermined bits is readable as a command having a corresponding predetermined function.

With regard to claim 8, it is clear that the predetermined function corresponding to the read/write command is performed in response to the predetermined bits.

With regard to claim 9, it is clear that the predetermined function read/write produces a result, and wherein the processor is configured to receive the result.

With regard to claim 10, it is clear that the master or the processor the master unit M, capable of sending data to or read data from at least one slave unit S, is configured to provide the user-defined value such as address or read/write command in bits corresponding to the user-defined value during execution of the instruction, and wherein the predetermined function (read/write) uses the data to produce the result.

With regard to claim 11, it is clear that result from read/write, for example, is generated during execution of the instruction.

With regard to claim 12, that the predetermined function read/write produces a result, and wherein the processor is configured to receive the result during execution of the instruction comprising predetermined bits.

With regard to claim 20, Heberle discloses a data processing system, comprising: a processor (in Heberle, the master unit M, capable of sending data to or read data from at least one slave unit S, is readable as a processor) configured to execute an instruction comprising a user-defined address (at least a predetermined address in Heberle) and to provide the user-defined address during execution of the instruction; and a device (a slave S) comprising an addressable register (register storing the address of the slave), wherein the device (slave S) is coupled to receive the user-defined address (at least a predetermined address in Heberle) and configured to access the addressable register (register storing the slave address) in response to the user-defined address.

With regard to claim 21, it is clear that the slave S is readable as a peripheral device comprising a register storing the address of the slave.

With regard to claim 22, it is clear that the slave S comprises at least a serial port.

With regard to claim 23, it is clear that the addressable register storing the predetermined address of the slave is at least one of a control register, a status register, and a data register.

With regard to claim 24, the master and slave controllers are readable as the data management unit coupled between the slave S and the master M and configured to receive read data from the peripheral device and to provide the read data to the master/processor. See also claims 14 and 15.

Claims 35-40, 41-43 are rejected under 35 U.S.C. 102(b) as being anticipated by Aatresh et al. (Aatresh, 5,469,544).

Aatresh discloses a method for obtaining a value stored in an addressable register (addressable memory registers, as in common memory architecture or address of an external device, see at least column 8, line 40 to column 9, line 14), comprising: driving an address of the addressable register on a plurality of address signal lines of a bus (in Aatresh, a microprocessor drives an address onto an address bus), and an asserted read control signal on a read control signal line of the bus (memory read/write operation using read/write control signals), during a first stage of an instruction execution pipeline (address pipelining); and receiving the value via a plurality of data signal lines of the bus when a corresponding ready signal driven on a ready signal line (RDY# in Aatresh) of the bus is asserted during a second stage of the instruction execution pipeline subsequent to the first stage (in Aatresh, next address to be strobed and decoded while the data corresponding to the current address is being transferred).

With regard to claim 36, since the memory/external device is operable in a pipelined manner, and with the use of RDY#, it is clear that a time period between the first and second stages of the instruction execution pipeline is extended in the event of a stall condition.

With regard to claim 37, Aatresh discloses a method for providing a value stored in an addressable register (memory addressable register, as in common memory architecture or address of an external device, see at least column 8, line 40 to column

Art Unit: 2111

9, line 14), comprising: receiving an address driven on a plurality of address signal lines of a bus when a read control signal driven on a read control signal line of the bus is asserted during a first stage of an instruction execution pipeline (in Aatresh, a microprocessor drives an address onto an address bus comprising a plurality of address signal lines); and if the address is an address of the addressable register, driving the contents of the addressable register on a plurality of data signal lines of the bus (if the address specified by the processor matches the address of the memory or address of an external device, then data is transmitted via the data bus comprising a plurality of data signal lines in response to the request of the processor), and an asserted ready signal (RDY# in Aatresh) on a ready signal line of the bus, during a second stage of the instruction execution pipeline subsequent to the first stage.

With regard to claim 38, Aatresh discloses a method for storing a value in an addressable register, comprising: driving an address of the addressable register (memory addressable register, as in common memory architecture or address of an external device, see at least column 8, line 40 to column 9, line 14) on a plurality of address signal lines of a bus, and an asserted write control signal on a write control signal line of the bus (memory read/write operation using read/write control signals), during a first stage of an instruction execution pipeline; and driving the value to be stored in the addressable register on a plurality of data signal lines of the bus, and an asserted ready signal (RDY# in Aatresh) on a ready signal line of the bus, during a second stage of the instruction execution pipeline subsequent to the first stage

(pipelining address is used in Aatresh, and in Aatresh, next address to be strobed and decoded while the data corresponding to the current address is being transferred).

With regard to claim 39, since the memory/external device is operable in a pipelined manner, and with the use of RDY#, it is clear that a time period between the first and second stages of the instruction execution pipeline is extended in the event of a stall condition.

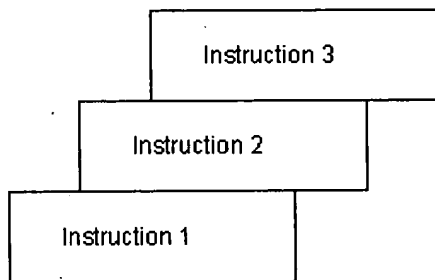
With regard to claim 40, see discussion above regarding claims 35-39.

With regard to claim 41, see above discussion. Note also that pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time. A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

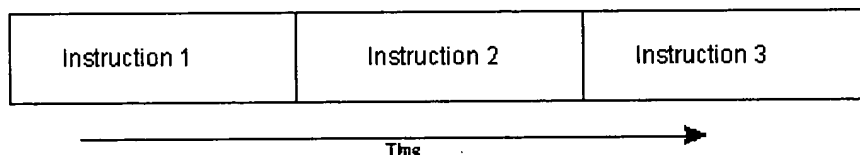
1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register

Pipelined and non-pipelined implementations are illustrated below:

Pipelined Implementation



Non-Pipelined Implementation



Note also that it is inherent that memory operations include read modify write (RMW). When you perform any operation on a register that changes its contents, the master/processor must first read the register, then it performs the operation on the number it has just read and finally writes the number back to the register. In addition, since the memory/external device is operable in a pipelined manner, and with the use of RDY# between stages or phases, it is clear that the address/read phase, return read, modify read value, and write phase are performed in different stages/phases in a pipelining architecture.

With regard to claim 42, since the memory/external device is operable in a pipelined manner, and with the use of RDY#, it is clear that a time period between the stages of the instruction execution pipeline is extended in the event of a stall condition.

With regard to claim 43, since the memory/external device is operable in a pipelined manner, and with the use of RDY#, it is clear that a time period between the stages of the instruction execution pipeline is extended in the event of a stall condition.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

With regard to claim 3, as discussed above, Heberle discloses the claimed invention including the use of a bus having a single signal line. Heberle does not disclose the use of a bus having a plurality of signal lines. However, the use a bus having a plurality of signal lines is old and well-known as evidenced from at least Antles, II, Bridges et al., or Aatresh, wherein a master and a plurality of I/O/slave devices are communicated over bus having a plurality of signal lines. It would have been obvious to one of ordinary skill in the art at the time the invention was made to employ a bus

Art Unit: 2111

having a plurality of signal lines, since the Examiner takes Official Notice that the use of a bus having a plurality of signal lines is old and well-known as evidenced from at least Antles, II, Bridges et al., or Aatresh, and using a bus having a plurality of signal lines for increasing data transfer in Heberle only involves ordinary skill in the art.

Claims 13 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

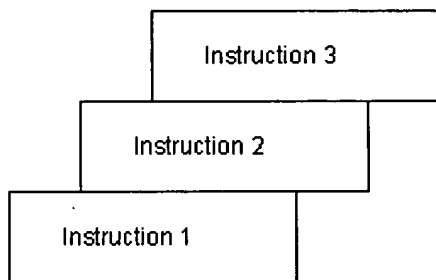
As discussed above, Heberle discloses the claimed invention. Heberle does not disclose that the master/processor executes the instruction via a plurality of pipeline stages. However, the use of pipeline stages is old and well-known as evidenced from the acknowledged background of the invention, which clearly states that “[m]any modern processors employ a technique called pipelining to execute more software program instructions (instructions) per unit of time. In general, processor execution of an instruction involves fetching the instruction (e.g., from a memory system), decoding the instruction, obtaining needed operands, using the operands to perform an operation specified by the instruction, and saving a result. In a pipelined processor, the various steps of instruction execution are performed by independent units called pipeline stages. In the pipeline stages, corresponding steps of instruction execution are performed on different instructions independently, and intermediate results are passed to successive stages. By permitting the processor to overlap the executions of multiple instructions, pipelining allows the processor to execute more instructions per unit of time. In general, a ‘scalar’ processor issues instructions for execution one at a time, and

a 'superscalar' processor is capable of issuing multiple instructions for execution at the same time. A pipelined scalar processor concurrently executes multiple instructions in different pipeline stages; the executions of the multiple instructions are overlapped as described above. A pipelined superscalar processor, on the other hand, concurrently executes multiple instructions in different pipeline stages, and is also capable of concurrently executing multiple instructions in the same pipeline stage." As a matter of fact, pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time. A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

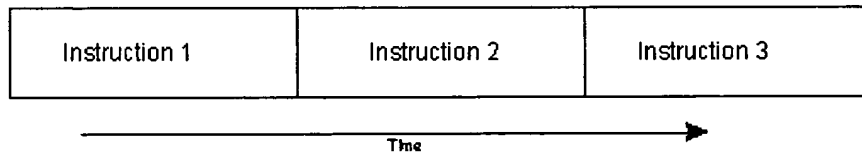
1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register

Pipelined and non-pipelined implementations are illustrated below:

Pipelined Implementation



Non-Pipelined Implementation



It would have been obvious to one of ordinary skill in the art at the time the invention was made to use “pipeline stages” in executing the instructions, since the use of such “pipeline stages” is old and well known as explained above, and using the same in Heberle, for the purpose of increasing the number of instructions executed, only involves ordinary skill in the art.

Claims 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

With regard to claims 15-19, as discussed above, Heberle discloses the claimed invention including the use of a bus having a single signal line. Heberle does not disclose the use of a bus having a plurality of signal lines. However, the use a bus having a plurality of signal lines is old and well-known as evidenced from at least Antles, II, Bridges et al., or Aatresh, wherein a master and a plurality of I/O/slave devices are communicated over bus having a plurality of signal lines. It would have been obvious to one of ordinary skill in the art at the time the invention was made to employ a bus having a plurality of signal lines, since the Examiner takes Official Notice that the use of a bus having a plurality of signal lines is old and well-known as evidenced from at least Antles, II, Bridges et al., or Aatresh, and using a bus having a plurality of signal lines for increasing data transfer in Heberle only involves ordinary skill in the art.

Claims 25-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

With regard to claims 25-27, as discussed above, Heberle discloses the claimed invention. Heberle does not disclose the use of interrupts and interrupt controller (claim 25), and interrupt based on priority (claim 26). However, the use of interrupts and priority interrupt controller is old and well-known as evidenced from at least the background of the claimed invention or Antles, II. The acknowledged background states that the term "interrupt request signal," or simply "interrupt signal," refers to a control

signal which indicates a high-priority request for service. For example, a peripheral device connected to a processor may assert an interrupt signal when ready to transmit data to the processor, or to receive data from the processor. It is noted that an interrupt signal generated external to a processor may not be synchronized with a clock signal of the processor. The two general categories of types of interrupt signals are 'non-maskable' and 'maskable'. The typical processor described above also has a non-maskable interrupt (NMI) terminal for receiving an NMI signal, and a maskable interrupt (IRQ) terminal for receiving an IRQ signal. The NMI signal is typically asserted when a catastrophic event has occurred or is about to occur. Examples of non-maskable interrupts include bus parity error, failure of a critical hardware component such as a timer, and imminent loss of electrical power. In general, maskable interrupts are lower-priority requests for service that need not be tended to immediately. Maskable interrupts may be ignored by the processor under program control. A request for service from a peripheral device which is ready to transmit data to a processor, or receive data from the processor, is an example of a maskable interrupt. An interrupt controller (e.g., a programmable interrupt controller or PIC) connected to the processor typically receives maskable interrupt requests from devices connected to the processor, and prioritizes the interrupt requests. When a processor receives an interrupt, application program execution stops, the contents of certain critical registers are saved (e.g., the internal state of the processor is saved), and internal control is transferred to an interrupt service routine (e.g., an interrupt handler) which corresponds to the type of interrupt received. In the case of a maskable or non-maskable interrupt, the interrupt controller typically

identifies the interrupt to be serviced. In a vectored interrupt system, the interrupt controller typically provides a number or instruction address assigned to the interrupt to an instruction sequencing module of the processor (e.g., during an interrupt acknowledge operation). A non-maskable interrupt is typically assigned a specific interrupt number. The processor uses the interrupt number as an index into the interrupt vector table to obtain the address of the appropriate interrupt service routine. When the interrupt service routine is completed, the saved contents of the critical registers are restored (e.g., the state of the processor is restored), and the processor resumes application program execution at the point where execution was interrupted." Antles discloses the use of a plurality of interrupts line EI 1-4 from the I/O slave devices, and a means for resolving conflicts between interrupts (interrupt controller). Note also that it is inherent that the processor comprises plurality of registers and logic for controlling the registers, wherein the logic is configured to operate the registers in response to the interrupt signals having different priority.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to employ interrupts and priority interrupt controller in Heberle, since the Examiner takes Official Notice that the use of interrupts and priority interrupt controller is old and well-known as evidenced from at least the background of the claimed invention or Antles, and using interrupts and priority interrupt controller in Heberle only involves ordinary skill in the art.

Claims 28 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

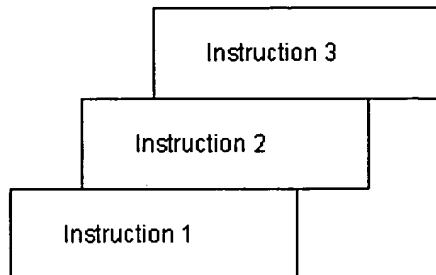
With regard to claims 28 and 30, as discussed above, Heberle discloses the claimed invention. Heberle does not disclose that the master/processor executes the instruction via a plurality of pipeline stages. However, the use of pipeline stages is old and well-known as evidenced from the acknowledged background of the invention, which clearly states that "[m]any modern processors employ a technique called pipelining to execute more software program instructions (instructions) per unit of time. In general, processor execution of an instruction involves fetching the instruction (e.g., from a memory system), decoding the instruction, obtaining needed operands, using the operands to perform an operation specified by the instruction, and saving a result. In a pipelined processor, the various steps of instruction execution are performed by independent units called pipeline stages. In the pipeline stages, corresponding steps of instruction execution are performed on different instructions independently, and intermediate results are passed to successive stages. By permitting the processor to overlap the executions of multiple instructions, pipelining allows the processor to execute more instructions per unit of time. In general, a 'scalar' processor issues instructions for execution one at a time, and a 'superscalar' processor is capable of issuing multiple instructions for execution at the same time. A pipelined scalar processor concurrently executes multiple instructions in different pipeline stages; the executions of the multiple instructions are overlapped as described above. A pipelined superscalar processor, on the other hand, concurrently executes multiple instructions in different

pipeline stages, and is also capable of concurrently executing multiple instructions in the same pipeline stage.” As a matter of fact, pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time. A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

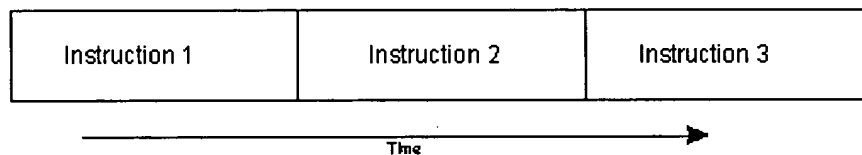
1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register

Pipelined and non-pipelined implementations are illustrated below:

Pipelined Implementation



Non-Pipelined Implementation



With regard to claim 30, note that it is clearly inherent that the address and read control signals occur during a first one of the pipeline stages, and read data obtained from the addressable register occurs during a second one of the pipeline stages.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use "pipeline stages" in executing the instructions, since the use of such "pipeline stages" is old and well known as explained above, and using the same in Heberle, for the purpose of increasing the number of instructions executed, only involves ordinary skill in the art.

Claims 29 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle.

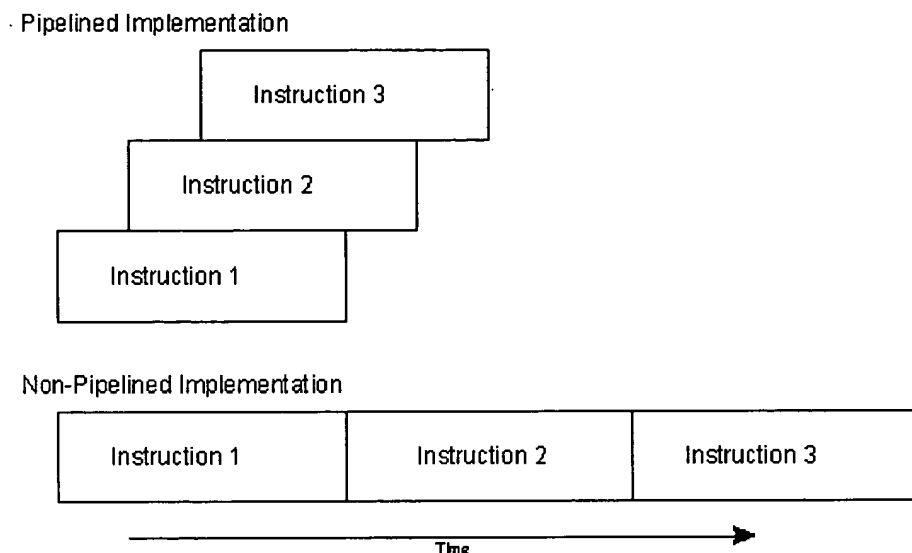
With regard to claims 29 and 31, as discussed above, Heberle discloses the claimed invention. Heberle does not disclose that the master/processor executes the instruction via a plurality of pipeline stages. However, the use of pipeline stages is old and well-known as evidenced from the acknowledged background of the invention, which clearly states that “[m]any modern processors employ a technique called pipelining to execute more software program instructions (instructions) per unit of time. In general, processor execution of an instruction involves fetching the instruction (e.g., from a memory system), decoding the instruction, obtaining needed operands, using the operands to perform an operation specified by the instruction, and saving a result. In a pipelined processor, the various steps of instruction execution are performed by independent units called pipeline stages. In the pipeline stages, corresponding steps of instruction execution are performed on different instructions independently, and intermediate results are passed to successive stages. By permitting the processor to overlap the executions of multiple instructions, pipelining allows the processor to execute more instructions per unit of time. In general, a ‘scalar’ processor issues instructions for execution one at a time, and a ‘superscalar’ processor is capable of

issuing multiple instructions for execution at the same time. A pipelined scalar processor concurrently executes multiple instructions in different pipeline stages; the executions of the multiple instructions are overlapped as described above. A pipelined superscalar processor, on the other hand, concurrently executes multiple instructions in different pipeline stages, and is also capable of concurrently executing multiple instructions in the same pipeline stage.” Note also that the use of ready signal between stages or phases of a pipeline is old and well-known as evidenced from Aatresh et al. As a matter of fact, pipelining, a standard feature in RISC processors, is much like an assembly line.

Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time. A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register

Pipelined and non-pipelined implementations are illustrated below:



It would have been obvious to one of ordinary skill in the art at the time the invention was made to use “pipeline stages” (having ready signal asserted between stages or phases) in executing the instructions, since the use of such “pipeline stages” is old and well known as explained above, and using the same in Heberle, for the purpose of increasing the number of instructions executed, only involves ordinary skill in the art. Note that it is clearly inherent that the address and read control signals occur during a first one of the pipeline stages, and read data obtained from the addressable register occurs during a second one of the pipeline stages.

Claims 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Heberle above, and further in view of the following.

With regard to claims 32-34, as discussed above, Heberle discloses the claimed invention. Heberle does not disclose that the master/processor executes the instruction via a plurality of pipeline stages. However, the use of pipeline stages is old and well-known as evidenced from the acknowledged background of the invention, which clearly states that “[m]any modern processors employ a technique called pipelining to execute more software program instructions (instructions) per unit of time. In general, processor execution of an instruction involves fetching the instruction (e.g., from a memory system), decoding the instruction, obtaining needed operands, using the operands to perform an operation specified by the instruction, and saving a result. In a pipelined processor, the various steps of instruction execution are performed by independent units called pipeline stages. In the pipeline stages, corresponding steps of instruction execution are performed on different instructions independently, and intermediate results are passed to successive stages. By permitting the processor to overlap the executions of multiple instructions, pipelining allows the processor to execute more instructions per unit of time. In general, a ‘scalar’ processor issues instructions for execution one at a time, and a ‘superscalar’ processor is capable of issuing multiple instructions for execution at the same time. A pipelined scalar processor concurrently executes multiple instructions in different pipeline stages; the executions of the multiple instructions are overlapped as described above. A pipelined superscalar processor, on the other hand, concurrently executes multiple instructions in different pipeline stages,

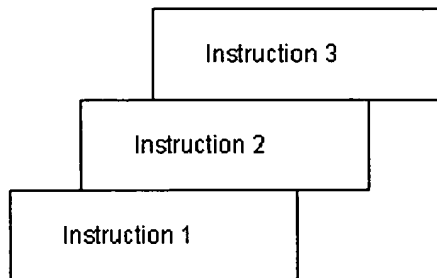
and is also capable of concurrently executing multiple instructions in the same pipeline stage." Note also that the use of ready signal between stages or phases of a pipeline is old and well-known as evidenced from Aatresh et al. As a matter of fact, pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time. A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register

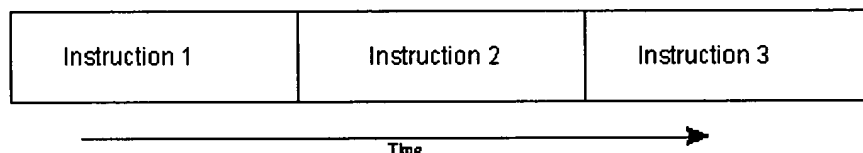
Note also that it is inherent that memory operations include read modify write (RMW). When you perform any operation on a register that changes its contents, the master/processor must first read the register, then it performs the operation on the number it has just read and finally writes the number back to the register.

Pipelined and non-pipelined implementations are illustrated below:

Pipelined Implementation



Non-Pipelined Implementation



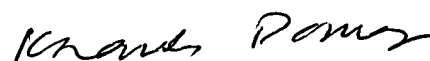
It would have been obvious to one of ordinary skill in the art at the time the invention was made to use "pipeline stages" (having ready signal asserted between stages or phases) in executing the instructions, since the use of such "pipeline stages" is old and well known as explained above, and using the same in Heberle, for the purpose of increasing the number of instructions executed, only involves ordinary skill in the art. Note that it is clearly inherent that the address and read control signals occur during a first one of the pipeline stages, and read data obtained from the addressable register occurs during a second one of the pipeline stages.

Art Unit: 2111

Claims 24-42 have been renumbered as claims 25-43.

U.S. Patent Nos. 6,081,860 to Bridges et al., 5,555,425 to Zeller et al., 6,256,693 to Platko, 4,942,519 to Nakayama, 5,327,121 to Antles II, 6,772,254 to Hoffman et al., and 5,729,703 to Onn et al. are cited as relevant art.

Any inquiry concerning this communication should be directed to Khanh Dang at telephone number 571-272-3626.

A handwritten signature in black ink, appearing to read "Khanh Dang", written in a cursive style.

**Khanh Dang
Primary Examiner**